1.0 INTRODUCTION

1.0 INTRODUCTION

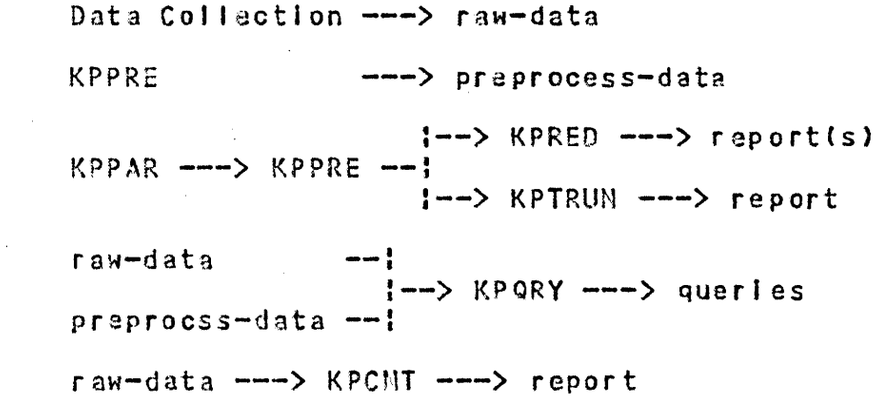
The 800 series machines have a performance monitoring facility (PMF) built into the hardware. Various types of data are made available for monitoring system performance. This introduction will not deal with the details of what is available or the details of how to collect the data.

The collection is initiated from the system console or by using the PMF procedure described below for systems which, at this time, are running in dual state. Keypoint instructions have been inserted at the gated entries and exits of significant procedures in the operating system (OS) and product set (PS). There may also be other keypoints inserted for data collection or other purposes. In addition, some processor state data is available. This user guide is directed towards the users of the keypoint data.

The keypoint data must be analyzed to obtain useful information. The set of programs in this user guide is directed toward describing the tools available to analysts for transforming data into information.

The keypoint collection is done from the operator console while NOSVE is running the job(s) to be analyzed. The output is the raw keypoints data file (KPTS). This is run each time a job(s) is to be analyzed (see PRFERS for full details).

The following is an example of the logical sequence and interaction of programs and data:



1.0 INTRODUCTION

PMF is run to initiate the PMF data collection programs for the purpose of collecting keypoints or counter information.

KPPAR is run to produce the PARSE file. This process modifies the keypoint data on the common decks into a form suitable for the preprocessor (KPPRE). This process is rerun only if the common decks are changed. The output, PARSFL may be used by all of the PMF users (see PRFERS for full details).

KPPRE transforms the raw data into a common format for the subsequent analysis programs. It can do some editing of the file using the directives from the KLUG file. It can also select portions of the raw data file using the start (STR) and stop (STP) directives in the parameter input. The output from the preprocessor are the preprocessed keypoint file, the descriptor file and several reports (see PRFERS for full details).

KPQRY allows scanning of either the raw keypoints data file or the preprocessed data file to determine some of the data in the keypoint files. This is intended to make quick, ad hoc inquiries about the nature of the keypoints (see PRFERS for full details).

KPRED uses the data from the descriptor file and the preprocessed keypoints to make more extensive, but predefined, reports of the contents of the keypoint runs (see PRFERS for full details).

KPTRUN is a process (Fig. 2) for analyzing the entries and exits from gated keypoints. It reconstructs the stack to determine the execution times within each gated keypoint. It is composed of 3 CYBIL programs and 2 sorts. CALLAN reads the preprocessed keypoint file and builds the network of caller-callee relationships that exist on the keypoint file. To keep the memory requirements low, and to decrease the processing time, the networks are are dumped periodically (every 3000 keypoints) to an intermediate file. The sort program simply bunches all the like keypoints from the interval dumps together for input to the COMBINE program. The COMBINE program sums the data from all the keypoints with the same parent and/or child into one record. It outputs these records together with the keypoint descriptions on a file in ascii format on the file specified by the FIRST parameter. Each record contains a parent and usually a child in ascii format. The processing to this point needs only to be done once for any given keypoint file. If reports are desired in several different sort orders, the NEWSORT parameter allows the processing to start at this point for the second and

subsequent reports. The intermediate file is now sorted using the SORTKEY parameters specified to order the records for the desired report. Finally, the PRINT process reads this ordered file and splits the combined caller-callee records into records suitable for printing.

2.0 PMF

2.0 PME

This procedure is used to initiate the PMF counters or the keypoint collection. It is also used to terminate a PMF session. If PMF has already been initiated for collecting counter data, the counters will be reset and the counter data already collected will be lost. If PMF has already been initiated for the purpose of collecting keypoints, then the already running collection will continue, and the present initiation will be aborted. If the PMF is for the purpose of terminating a run (END) then the presently running PMF will be terminated.

2.1 PME PARAMETERS

TY — This parameter sets the type of the PMF call. The 3 allowable values are BEGIN, COLLECT, and END. BEGIN is used to initiate PMF when only counter information is to be collected. COLLECT is used to initiate keypoint collection runs. END is used to terminate either type of run.
EXAMPLE TY=COLLECT DEFUALT=END

MI — Machine Identification. This paramter tells the PP program what type of machine it is running on. Allowable values are S1, S2, OR S3.
EXAMPLE MI=S1 DEFAULT=S2

IF — Input File. This is an optional parameter. If only TY and MI are to be specified, a separate input file is not required. If the values for the counter collection are to be modified, then these parameters should be placed in the input file. If an input file is used, be sure to include the values for TYPE and MID as the values from the TY and MI above will not be used.
EXAMPLE IF=INPF. DEFAULT none.

OF — Output File. If specified, the direct acess file specified will be used for the PMF output unless TPF (tape output) is specified. If neither OF nor TPF is specified, the output will be stored on a file named KPTS (Direct Access).
EXAMPLE OF=COLLFIL DEFAULT= KPTS.

2.0 PMF
2.1 PMF PARAMETERS

TPF — If the raw input data is on tape, TPF is used to specify the VSN(s) of the tape(s) on which the data is stored. If the VSN(s) have leading 0s the VSN must be enclosed in single quote marks. If the data is spread across more than one reel, the list of VSNs must be enclosed in parenthesis. Up to 10 reels of tape may be specified. Tapes must be labeled.
Example TPF=('000123',123456). DEFAULT none.

DE — Tape density for TPF.
Example DE=PE. DEFAULT GE.

LIB — Name of file where the object code for OCPKPNOS is found. Ordinarily this parameter is obtained from the 'toollib' parameter in the user's PROFILE. This parameter would ordinarily be used for testing a new version of the collection routine, or for use of a specially modified version of the collection routine.
Example LIB=MYLIB. DEFAULT XULIB.

LIBUN — User number where file specified by LIB (above) resides. Ordinarily this parameter is obtained from the 'toolact' parameter in the user's PROFILE.
Example LIBUN=UN12. DEFAULT KEYPERF.

2.2 TOGGLING KEYPOINT COLLECTION IN THE PRE TOOL

PMF hardware start/stop and keypoint collection on/off are under the control of class 15 keypoints in the following manner:

CLASS	KCD = KPT.	CODE EXPLANATION
15	0	start PMF hardware
15	1	stop PMF hardware
15	2	turn keypoint collection on
15	3	turn keypoint collection off

Thus, once PPKPNOS is loaded into the PP, PMF hardware is started by the first (class 15) keypoint encountered having a Keypoint Code (KCD) of 0. However, keypoint collection will

not begin until the first (class 15) keypoint having a KCD = 2 is encountered. Collection is temporarily stopped when a (class 15 keypoint having) KCD=3 is encountered. Collection can be started and stopped as many times as desired by alternately issuing KCD=2 and KCD=3. Collection is irrevocably terminated by a KCD=1, which stops the PMF hardware and causes PPKPNOS to drop out.

The following example symbolically illustrates the use of this feature. A program has five procedures, all of them keypointed. (They may be called in any order.) However, it is only desired to collect keypoint data from procedures B and D. Class 15 keypoints are inserted in the program at appropriate points to achieve the intended effect.

2.0 PMF
2.2 TOGGLING KEYPOINT COLLECTION IN THE PRF TOOL

POSSIBLE USE OF ON/OFF COLLECT FEATURE

```
        main program start
        keypoint,class=15,code=0              (initiate PMF hardware)
          .
          .
          .
        +------
        |  procedure entry
        |
A       |  (main body)
        |
        |  procedure exit
        +------

        +------
        |  procedure entry
        |  keypoint,class=15,code=2           (start keypoint collection)
B       |  (main body)
        |  keypoint,class=15,code=3
        |  procedure exit                     (stop  keypoint collection)
        +------

        +------
        |  procedure entry
        |
C       |  (main body)
        |
        |  procedure exit
        +------

        +------
        |  procedure entry
        |  keypoint,class=15,code=2           (start keypoint collection)
D       |  (main body)
        |  keypoint,class=15,code=3           (stop  keypoint collection)
        |  procedure exit
        +------

        +------
        |  procedure entry
        |
E       |  (main body)
        |
        |  procedure exit
        +------
          .
```

2.0 PMF
2.2 TOGGLING KEYPOINT COLLECTION IN THE PRF TOOL

Throughout this discussion, we implicitly refer only to class 15 keypoints.

1 PMF hardware will actually start upon encountering any class 15 keypoint, regardless of its keypoint code. However, for the purposes of clarity and convention, KCD=0 should always be used. Also, do not attempt to start PMF with KCD=1, since PMF will start and immediately stop again.

2 When KCD=1 is encountered, PMF is stopped by writing byte zero. Any data contained in byte zero will be lost, since writing byte zero also causes it to be cleared (qv. MIGDS 2.11.2).

3 All class 15 keypoints, regardless of whether collection is on or off, are collected. KCD=1 (STOP PMF) is the only exception. Therefore, always issue a KCD=3 before issuing a KCD=1.

Normal exits from a procedure to the end of a program are then distinguished from abnormal exits from the middle of a procedure to the end of program. The former leaves two KCD=3 keypoints at the end of the data collection file; the latter leaves only one. (See the symbolic programming example two pages previous.)

4 Once PMF hardware is started, further occurrences of KCD=0 will not cause PMF hardware to restart.

5 Issuing either a KCD=2 while collection is on or a KCD=3 while collection is off, will have no effect.

6 Only the last 2 bits of the KCD are sensed by PPKPNOS. For example, KCD=A will act in identical manner to KCD=2.

7 At present, the time elapsed between the issuing of a KCD=3 and the issuing of the next KCD=2 is unknown. Subtracting the time stamp of the KCD=3 from that of the KCD=2 does not necessarily give the elapsed time,

```
      .
      .
keypoint,class=15,code=3      (recommended, but not required)
keypoint,class=15,code=1      (terminate PMF hardware and drop PP)
main program end
```

RUNNING THE COLLECTOR

The on/off collection feature does not affect the CP part of the keypoint collection program (CPKPNOS). However, PMF control byte (byte 1) bits 8 and 9 must be set and bit 10 must be clear. Currently, the default is set for all three bits. CTL=192 in the input file will reset the control byte in the proper manner. The PMF hardware will start upon encountering the first class 15 keypoint with KCD=0 and will only stop upon encountering a class 15 keypoint with KCD=1.

Here is an example of the commands used to run keypoint collection on an S1:

```
SES.PMF TY=COLLECT IF=INFILE
```

where INFILE contains:

```
TYPE=COLLECT
MID= 1
CTL=192 (reset control byte from default)
SCO=0 (disables PMF stop-on-counter-overflow)
```

The following parameters must be left at their default values:

SMF on/off collect feature can't be used at same time as sampling
SKC starts PMF hardware upon recognition of a class 15 keypoint

If the sampling option is selected, the toggled keypoint collection feature is disabled.

NOTES ON COLLECTOR OPERATION.

since the PMF microsecond timer could have wrapped around an undetermined number of times between issuing the KCD=3 and issuing the KCD=2.

3.0 KPPRE

3.0 KPPRE

This program transforms the raw data from the data collection file into a common format for use by the rest of the PMF programs. It also produces some reports about the keypoint data. Some editing of the data is also possible through use of the KLUG file. Debug data is written to a file 'DEBGFL'. If this file exists in the user's file space, the data is appended to the current contents of DEBGFL, otherwise a file 'DEBGFL' is created.

3.1 KPPRE PARAMETERS

TPF — If the raw input data is on tape, TPF is used to specify the VSN(s) of the tape(s) on which the data is found. If the VSN(s) have leading 0s the VSN must be enclosed in single quote marks. If the data is spread across more than one reel, the list of VSNs must be enclosed in parenthesis. Up to 10 reels of tape may be specified. Tapes must be labeled.
Example TPF=('000123',123456). DEFAULT none.

DEI — Tape density for TPF.
Example DEI=PE. DEFAULT GE.

DTF — If the raw keypoint data is on mass storage, DTF specifies the file name in which the data is stored. The file may be local, direct access, or even indirect access although the last alternative is usually impractical.
Example DTF=DATAFL. DEFAULT DATAFL.

DSF — This parameter specifies the file where the keypoint descriptors will be stored.
Example DSF=DESCFL. DEFAULT none, this is a required parameter.

PRF — Parsed keypoints common decks file. This file is the output file from KPPAR. If the PRF file is not found, KPPRE runs KPPAR to create one.
Example PRF=PARSFL. DEFAULT PARSFL.

KLF — This parameter allows specification of a file containing information about keypoints which should be ignored or erroneous (usually because

3.0 KPPRE
3.1 KPPRE PARAMETERS

they are improperly installed).
Example KLF=KLUGFL. DEFAULT KLUGFL.

KPF — This parameter specifies the file on which the preprocessed keypoints will be stored.
Example KPF=KPTDFL. DEFAULT none, this is a required parameter.

SVF — This parameter allows the VSN(s) of an output tape on which the preprocessed keypoints will be stored. Note that SVF specifies only the VSN(s), KPF is still required, although KPF is now the LFN of the tape. Up to 10 VSN(s) may be specified.
Example SVF=('000123','000456'). DEFAULT none.

DEO — Tape density for SVF.
Example DEO=PE. DEFAULT GE.

TRF — This parameter specifies the file name for the trace file which stores diagnostics and data about the keypoints. If the file specified exists in the user's catalog, then the data is appended to that file. Otherwise, a file is created using the specified name.
Example TRF=TRACFL. DEFAULT TRACFL.

TKF — This parameter specifies a preprocessed keypoint file for a selected task.
Example TKF=TASK1. DEFAULT TASKFL.

SKP — Number of keypoints to skip at the beginning of the raw keypoint file.
Example SKP=1000. DEFAULT 0.

MKP — Maximum number of keypoints to process from the raw keypoint data file.
Example MKP=2000. DEFAULT 99999999.

STR — Start class and keypoint. This parameter allows searching the raw keypoint data file to the point where a keypoint of a given class is found. The format is 'CLASS KEYPOINT'. Space between class and keypoint is required. See note on cycling in STP description.
Example STR='4 4000'. DEFAULT '-1 -1'.

STP — Stop class and keypoint. This parameter allows stopping the preprocessor after a given keypoint

has been found. Note, however, that collection will resume if another keypoint of STR class and keypoint is found, so that this start stop process is cyclical.
Example STP='4 4001'. DEFAULT '-1 -1'.

TSK — This parameter allows selection of only those keypoints corresponding to a given task to be preprocessed.
Example TSK=5. DEFAULT -1 (all tasks)

PRE — This is a toggle to produce a preprocessed keypoint file.
Example PRE=N. DEFAULT Y.

CDS — This is a toggle to produce a compressed keypoint descriptor file.
Example CDS=N. DEFAULT Y.

RDS — This is a toggle to replace the description into the PL.
Example RDS=Y. DEFAULT N.

FLT — This parameter specifies the type of input data on the keypoints file. Allowable types are HARD, SOFT, SIM
Example FLT=SOFT. DEFAULT HARD.

TRC — This parameter specifies whether trace output is to be generated.
Example TRC=Y. DEFAULT N.

DSC — This parameter specifies whether a full keypoint descriptor file is to be generated.
Example DSC=Y. DEFAULT N.

ID — This parameter allows specification of the UN of the workspace containing certain files. This allows a single copy of these files to exist, simplifying procedures when they are updated. The files accessed using the ID parameter are KLF, PRF and DTF.
Example ID=KEYPERF. DEFAULT user's own UN.

LIB — Name of file where the object code for XKPPRE is found. Ordinarily this parameter is obtained from the 'toollib' parameter in the user's PROFILE. This parameter would ordinarily be used for testing a new version of the

preprocessor, or for use of a specially modified version of the preprocessor.
Example LIB=MYLIB. DEFAULT XULIB.

LIBUN — User number where file specified by LIB (above) resides. Ordinarily this parameter is obtained from the 'toolact' parameter in the user's PROFILE.
Example LIBUN=UN12. DEFAULT KEYPERF.

4.0 KPTRUN

4.0 KPTRUN

This proc directs the processing of call analysis of the keypoint file from the primary data reduction program (KPPRE). The report from this proc may be sorted into different orders using the parameters in this proc. The title which appears on each page may be specified in this proc. Some processing limits may also be established.

4.1 PARAMETERS

TPF	Tape number or numbers containing preprocessed keypoints. If the VSN has leading zeros the VSN must be enclosed in string quotes ('). If the file is a multi-reel tape, then the list of tapes must be enclosed in parenthesis. Example TPF=('000123',543210). DEFAULT none; exactly one of TPF and DTF must be specified.
DE	Tape density for TPF. Example DE=PE. DEFAULT GE.
DTF	Disk file containing the preprocessed keypoints. Example DTF=DTF1. DEFAULT none; exactly one of TPF and DTF must be specified.
DSF	File on which the keypoint descriptors reside. This is a required parameter. Example DSF=DSF1. DEFAULT none.
REPORT	Permanent file name for the report file. If the file specified exists in the user's catalog, then the data is appended to that file. Otherwise, a file is created using the specified name. Example REPORT=REP1. DEFAULT REPORT.
TAPE1	Permanent file name for the debugging file. If the file specified exists in the user's catalog, then the data is appended to that file. Otherwise, a file is created using the specified name. Example TAPE1=DEB1. DEFAULT TAPE1.
FIRST	Name of permanent file where intermediate sort data is stored. Example FIRST=INTERM. DEFAULT FIRST.

4.0 KPTRUN
4.1 PARAMETERS

ID — If the the preprocessed keypoint data file is in another user's area this parameter must be specified.

COMMENT — A title for each page on the report may be specified using this parameter. The comment may be up to 50 characters long. The string is enclosed in single quotes (').

PRINT — Dispose a copy of the report to the printer.

NOEXIT — No EXIT card inserted in the job stream.

RECORDS — If you wish to process less than the entire file specify the number of records you wish to process as RECORDS=nnnnnn.

TOTAL — Process entire file without regard to task boundaries.

NEWSORT — This keyword allows you to obtain a report sorted in another order without running the analysis phase of the call analysis (the time consuming part). Use it, coding the same parameters as you used for the original run.

STR — This parameter allows skipping the preprocessed input file until a given class and keypoint is found.
Example STR='12 4004'

STP — This parameter allows termination of the call analysis run when a given class and keypoint are found on the preprocessed keypoint file.
Example STP='13 4004'.

The following are the allowable parameters for the SORTKEY list. They must be enclosed within parenthesis.

SORTKEY — This is the keyword for the various sort options. The SORTKEY options appear below.

- KEYPNT — Report is in increasing values of keypoints and class (All keypoints for a given class come first).
  Example SORTKEY=(KEYPNT)
- FUNCT — Report is in increasing values of the 2 letter section name. (The 2 letters in the

third column of the report)

COUNT	Report is in increasing values of the count of entries to the keypoint
VCOUNT	Report is sorted in increasing value of matched exit count for the keypoints
CPTIME	Report is sorted on increasing values of CP time in the procedure.
AVCP	Report is sorted on increasing values of average CP time (CPTIME/MATCHED COUNT)
RLTIME	Report is sorted in increasing values of Real time
AVRT	Report is sorted in increasing order of average real time in the procedure
CCPT	Report is sorted in order of increasing values of Child CP time
PAGES	Report is sorted in order of increasing number of page faults.
CPAGES	Report is sorted in order of increasing number of Child page faults.
DESCR	Report is sorted with descriptions in alphanumeric order.
CHKYPT	The children of each parent are sorted keypoint order.
CHCPTM	The children of each parent are sorted in order of increasing CP time.

Any of the SORTKEY parameters may be sorted in decreasing order by prefixing the appropriate parameter with the letter D. For instance, to sort such that the highest CP time is printed first, use SORTKEY=DCPTIME.

The order in which the SORTKEY parameters appear in the list determines which key is most significant. eg. SORTKEY=(DCPTIME,DRLTIM,KEYPNT,DCHCPT) will sort first on CPTIME (highest first), then on real time(descending), then on keypoint(ascending) and finally on child CP time(descending). It is suggested that when sorting in orders other than keypnt,

at least one of the minor keys should be KEYPNT, in order to have all the data concerning a keypoint to appear in the same part of the report.

If you want to sort on any other field that appears on the report all that is required is to define a field and a key in this proc. To find the column number where the field appears, attach the file determined by the FIRST parameter in a previous run. This file may be inspected with an editor, keeping in mind that the line length is about 300 ascii characters. Determine the column in which the desired sort parameter lies, and modify the NOTE lines defining fields and keys for SORP accordingly.

If you inspect this proc you will see that the file named by the FIRST parameter is saved as a direct access file. The first program, CALLAN uses the bulk of the processing time, so different sort options may be obtained very cheaply by starting the program with the COMBINE program. This file is used by the NEWSORT option to allow a report sorted in some other order. Any number of reports may be generated by running KPTRUN with the NEWSORT parameter and using the proper copy of the FIRST file.

5.0 KPPAR

5.0 KPPAR

This program reads the keypoint common decks and creates a binary file summarizing the keypoint descriptions, for use by KPPRE. It also checks the syntax of every description according to the language specification.

Any of the parameters defined in the SES module JOBPARM may be used. These include JOBTL, JOBFL, JOBCN, JOBPR, JOBUN, JOBPW, JOBFMLY, JOBCN, JOBPN, LOCAL, BATCHN, BATCH, DEFER, NODAYF, DAYFILE, AND DF. (See SES Procedure Writer's Guide, Appendix A)

5.1 KPPAR PARAMETERS

PRF — Parsed keypoints common decks file. This file is the output file from KPPAR.
Example PRF=PARSFL. DEFAULT PARSFL.

CDF — This specifies the name of the file containing the common decks. If no file is assigned, KPPAR will attempt to get the common decks using file KEYDESC in user number DEV1.
Example CDF=KEYD. DEFAULT KEYDESC.

ID — If the the preprocessed keypoint data file is in another user's area this parameter must be specified.

LIB — Name of file where the object code for XKPPAR is found. Ordinarily this parameter is obtained from the 'toollib' parameter in the user's PROFILE. This parameter would ordinarily be used for testing a new version of the preprocessor, or for use of a specially modified version of the preprocessor.
Example LIB=MYLIB. DEFAULT XULIB.

LIBUN — User number where file specified by LIB (above) resides. Ordinarily this parameter is obtained from the 'toolact' parameter in the user's PROFILE.
Example LIBUN=UN12. DEFAULT KEYPERF.

6.0 KPRED

6.0 KPRED

The keypoint data reduction program reduces keypoint data gathered from the NOS/VE operating system. Keypoints must be processed by the keypoint preprocessor previously described before they may be analyzed by KPRED.

Any of the parameters defined in the SES module JOBPARM may be used. These include JOBTL, JOBFL, JOBCN, JOBPR, JOBUN, JOBPW, JOBFMLY, JOBCN, JOBPN, LOCAL, BATCHN, BATCH, DEFER, NODAYF, DAYFILE, AND DF. (See SES Procedure Writer's Guide, Appendix A)

6.1 KPRED PARAMETERS

TPF — If the preprocessed data is on tape, TPF is used to specify the VSN(s) of the tape(s) on which the data is found. If the VSN(s) have leading 0s the VSN must be enclosed in single quote marks. If the data is spread across more than one reel, the list of VSNs must be enclosed in parenthesis. Up to 10 reels of tape may be specified. Tapes must be labeled. This in an output from KPPRE.
Example TPF=('000123',123456). DEFAULT none.

DE — Tape density for TPF.
Example DE=PE. DEFAULT GE.

DTF — If the preprocessed data is on mass storage, DTF specifies the file name in which the data is stored. If DTF is not found, KPRED runs KPPRE to create one. The file may be local, direct access, or even indirect access although the last alternative is usually impractical. This is an output from KPPRE.
Example DTF=DATAFL. DEFAULT DATAFL.

DSF — This parameter specifies the file where the keypoint descriptors are found. This is an output from KPPRE.
Example DSF=DESCFL. DEFAULT none, this is a required parameter.

DDF — This parameter allows specification of an input file for reducing the processing during data

statistics analysis. The file format is:
Class Keypoint D (for delete), this means the data field will not be recorded for that keypoint.

RDF — This parameter specifies the lfn of the KPRED output file.
Example RDF=OUT. DEFAULT OKPRED.

DGF — This parameter specifies the lfn of the KPRED debug file.
Example DGF=DEBUG. DEFAULT DBGF.

SMF — This parameter allows specification of the KPRED summary file.
Example SMF=SUMMARY. DEFAULT SMYF.

TGL — This parameter allows setting of various program toggles. The permitted toggles are ABCDEJKLNRSTX.
Example TGL=BCX. DEFAULT CLNT.

TIT — This parameter allows specifying a title of up to 40 characters.
Example TIT='PMF RUN FOR FMU'. DEFAULT PMF RUN.

RNO — This parameter allows specifying a run number (max of 6 digits).
Example RNO=1. DEFAULT 99.

SKC — This parameter allows specifying the starting keypoint class for processing.
Example SKC=13. DEFAULT 0.

MER — This parameter specifies the number of errors to print. If the number of errors exceeds this number processing continues but the remaining errors are not printed.
Example MER=500. DEFAULT 1000

MDT — This parameter specifies the maximum number of data values to save per keypoint.
Example MDT=25. DEFAULT 100.

INT — This parameter allows the summary data to be dumped at some specified interval (in milliseconds). However, the follow on program for formatting the output has not been written yet. Therefore, do not use this parameter.

6.0 KPRED
6.1 KPRED PARAMETERS

Example INT=100. DEFAULT -1 (Ignore interval).

SCT — This parameter allows specification of which section number to print. This parameter works in concordance with the program toggle T (trace). This would be used if only one section of keypoints (ie., 1500-1550) is of interest (if the number is not the start of a section, modulo 50 division will be done such that for instance 1525 would print the keypoints 1500-1549).
Example SCT=1500. DEFAULT 9999 (all sections).

SKP — Number of keypoints to skip at the beginning of the raw keypoint file (DTF or TPF).
Example SKP=1000. DEFAULT 0.

MKP — Maximum number of keypoints to process from DTF.
Example MKP=2000. DEFAULT 99999999.

ID — This parameter allows specification of the UN of the workspace containing certain files. This allows a single copy of these files to exist, simplifying procedures when they are updated. The files accessed using the ID parameter are DDF, DTF and DSF.
Example ID=KEYPERF. DEFAULT user's own UN.

LIB — Name of file where the object code for XKPRED is found. Ordinarily this parameter is obtained from the 'toollib' parameter in the user's PROFILE. This parameter would ordinarily be used for testing a new version of the preprocessor, or for use of a specially modified version of the preprocessor.
Example LIB=MYLIB. DEFAULT XULIB.

LIBUN — User number where file specified by LIB (above) resides. Ordinarily this parameter is obtained from the 'toolact' parameter in the user's PROFILE.
Example LIBUN=UN12. DEFAULT KEYPERF.

7.0 KPQRY

The KPQRY program reads the keypoint file or the preprocessed file and extracts pertinent information according to the input-query parameters given. The first line of input specifies the type of file being analyzed. The next lines are in the form of a small query language. (See PRFERS for language syntax)

Any of the parameters defined in the SES module JOBPARM may be used. These include JOBTL, JOBFL, JOBCN, JOBPR, JOBUN, JOBPW, JOBFMLY, JOBCN, JOBPN, LOCAL, BATCHN, BATCH, DEFER, NODAYF, DAYFILE, AND DF. (See SES Procedure Writer's Guide, Appendix A)

DTF — If the raw data or preprocessed keypoints is on mass storage, DTF specifies the file name in which the data is stored. The file may be local, direct access, or even indirect access although the last alternative is usually impractical.
Example DTF=DATAFL. DEFAULT DATAFL.

TPF — Tape number or numbers containing the raw data or preprocessed keypoints. If the VSN has leading zeros the VSN must be enclosed in string quotes ('). If the file is a multi-reel tape, then the list of tapes must be enclosed in parenthesis.
Example TPF=('000123',543210). DEFAULT none.

DE — Tape density for TPF.
Example DE=PE. DEFAULT GE.

IF — This specifies the input parameter file where the KPQRY directives are found.
Example IF=PARM. DEFAULT none (interactive input).

OF — This specifies the file on which the output will be written. If the file specified exists in the user's catalog, then the data is appended to that file. Otherwise, a file is created using the specified name.
Example OF=OUT. DEFAULT none (interactive output).

7.0 KPQRY

LIB — Name of file where the object code for XKPQRY is found. Ordinarily this parameter is obtained from the 'toollib' parameter in the user's PROFILE. This parameter would ordinarily be used for testing a new version of the preprocessor, or for use of a specially modified version of the preprocessor.
Example LIB=MYLIB. DEFAULT XULIB.

LIBUN — User number where file specified by LIB (above) resides. Ordinarily this parameter is obtained from the 'toolact' parameter in the user's PROFILE.
Example LIBUN=UN12. DEFAULT KEYPERF.

8.0 KPCNT

8.0 KPCNT

This program reads in the register 22 from the data collected file and generates statistics according to machine type and selected counters.

Any of the parameters defined in the SES module JOBPARM may be used. These include JOBTL, JOBFL, JOBCN, JOBPR, JOBUN, JOBPW, JOBFMLY, JOBCN, JOBPN, LOCAL, BATCHN, BATCH, DEFER, NODAYF, DAYFILE, AND DF. (See SES Procedure Writer's Guide, Appendix A)

DTF — If the raw keypoint data is on mass storage, DTF specifies the file name in which the data is stored. The file may be local, direct access, or even indirect access although the last alternative is usually impractical.
Example DTF=DATAFL. DEFAULT DATAFL.

TPF — Tape number or numbers containing preprocessed keypoints. If the VSN has leading zeros the VSN must be enclosed in string quotes ('). If the file is a multi-reel tape, then the list of tapes must be enclosed in parenthesis.
Example TPF=('000123',543210). DEFAULT none; exactly one of TPF and DTF must be specified.

DE — Tape density for TPF.
Example DE=PE. DEFAULT GE.

CNF — This file will contain the counter statistics. If the file specified exists in the user's catalog, then the data is appended to that file. Otherwise, a file is created using the specified name.
Example CNF=OKPCNT. DEFAULT OKPCNT.

TIT — This parameter allows specifying a title of up to 40 characters.
Example TIT='PMF RUN FOR FMU'. DEFAULT PMF RUN.

LIB — Name of file where the object code for XKPCNT is found. Ordinarily this parameter is obtained from the 'toollib' parameter in the user's PROFILE. This parameter would ordinarily be used for testing a new version of the preprocessor, or for use of a specially modified version of the preprocessor.

8.0 KPCNT

Example LIB=MYLIB. DEFAULT XULIB.

LIBUN — User number where file specified by LIB (above) resides. Ordinarily this parameter is obtained from the 'toolact' parameter in the user's PROFILE.
Example LIBUN=UN12. DEFAULT KEYPERF.

9.0 INSTALLATION

9.0 INSIALLAIION

MATERIALS PMFPL
PMFCOR
XULIB

Add any local mods to PMFCOR, the file containing changes to the PMF programs.

Add the following statements to the PROFILE file.

```
\ PLIB= 'PMFPLIB'
\ TOOLLIB='XULIB'
\ TOOLACT= the PMF user name,ie., 'KEYPERF'
\ SEARCH = (PMFPLIB,the PMF user name)
```

After PMFPL, PMFCOR and XULIB have been loaded do

```
SES.GENCOMP PMFINST C=PMFCOR B=PMFPL
SES.REPPROC G=COMPILE B=PMFPLIB
SES.PMFINST ALL
```

The calls to GENCOMP and REPPROC need only be done when the PMF programs are being installed for the first time. From then on, only PMFINST needs to be called to install the particular modules which are to be altered.

To have a copy of this document do the following

```
SES.GENCOMP PMFUG B=PMFPL C=PMFCOR CF=PMFUG
SES.FORMAT PMFUG
```

9.1 PMFINSI

This procedure installs the various components of the PMF system. It installs the object code on a library file, and the SES procedures on a PLIB file. Prior to installation a file, PMFPL, containing the various modules must exist in the installation user number (UN). A second file, PMFCOR, containing the correction set must also be available. The PROFILE of the user must contain the definitions given above.

Any of the parameters defined in the SES module JOBPARM may be used. These include JOBTL, JOBFL, JOBCN, JOBPR, JOBUN, JOBPW, JOBFMLY, JOBCN, JOBPN, LOCAL, BATCHN, BATCH, DEFER, NODAYF, DAYFILE, AND DF. (See SES Procedure Writer's Guide, Appendix A)

9.0 INSTALLATION
9.1 PMFINST

Installation Parameters

CALLAN	Install the object for the call analysis portion of KPTRUN.
COMBINE	Install the object code for the program which combines the intermediate segments produced by CALLAN (Part of KPTRUN).
PRINT	Install the object code for the routine which prints the REPORT(s) (Part of KPTRUN)
CPKPNOS	Installs the CPU portion of the data collection routines.
PPKPNOS	Installs the PP portion of the data collection routines.
KPPRE	Install the object code for the Preprocessor.
KPPAR	Install the object code for the keypoint descriptions parser program.
KPRED	Install the object code for the primary data reduction program.
KPQRY	Install the object code for the KPQRY program.
KPCNT	Install the object code for the KPCNT program.

All of the above programs are stored on the file XULIB.

PRPMF	Install the SES proc for running the keypoint collection.
PRKPPRE	Install the SES proc for running KPPRE.
PRKPPAR	Install the SES proc for running KPPAR.
PRKPRED	Install the SES proc for running KPRED.
PRKPQRY	Install the SES proc for running KPQRY.
PRKPCNT	Install the SES proc for running KPCNT.
KPTRUN	Install the SES proc for running the call analysis programs.
PMFINST	Install the SES proc for installing the

installation program. This will usually be done only when the installation program has been modified in PMFCOR.

ALL — Calls for installation of all of the above programs and procedures.

The above procedures are installed on the SES proc library, PMFPLIB.

PLC — This parameter is used to change the name of PL170, if it is under a different name.

ID — This parameter is used to specify the account where PLC resides, if different than DEV1.

After PMFINST has completed batch execution, all binaries are found in XULIB. The PP data collection program is under the name PMF. The sequence of commands necessary to place the code in the system is

```
SES.GETMEM PMF PP B=XULIB
SYSEDIT,B=GROUP.
```

These commands must be entered from the system console.

9.2 PROGRAM MAINTENANCE

All PMF related material is stored on PMFPL. In order to maintain an orderly system of changing or correcting the programs and procedures, all changes are entered into a correction file. This file has two areas for entering changes. General changes are entered in the correction ID called PMFCOR. It is recognized that some sites will have special need to cater to their equipment or operational procedures. Accordingly there is a correction ID for site local modifications to be stored. If these changes are of general use, then they may be moved into the PMFCOR ID. Presently, the site local changes reflect the use of group encoding used on tape equipment rather than phase encoding. In addition, it may be desirable for some sites to change the default conditions on some of the procs and these may also go into the site local mods.

There are two methods of generating the corrections. One is to get the module from PMFPL applying the PMFCOR changes, and then changing the resultant module to fix the bug or introduce the change. Then after testing, make an SES GENCOR run to

produce the changes required to go into PMFCOR. The other way is to make a GENCOMP run with the SEQ parameter to get the line numbers, then enter the proper insert and delete entries in PMFCOR.

Table of Contents